FIG. 1

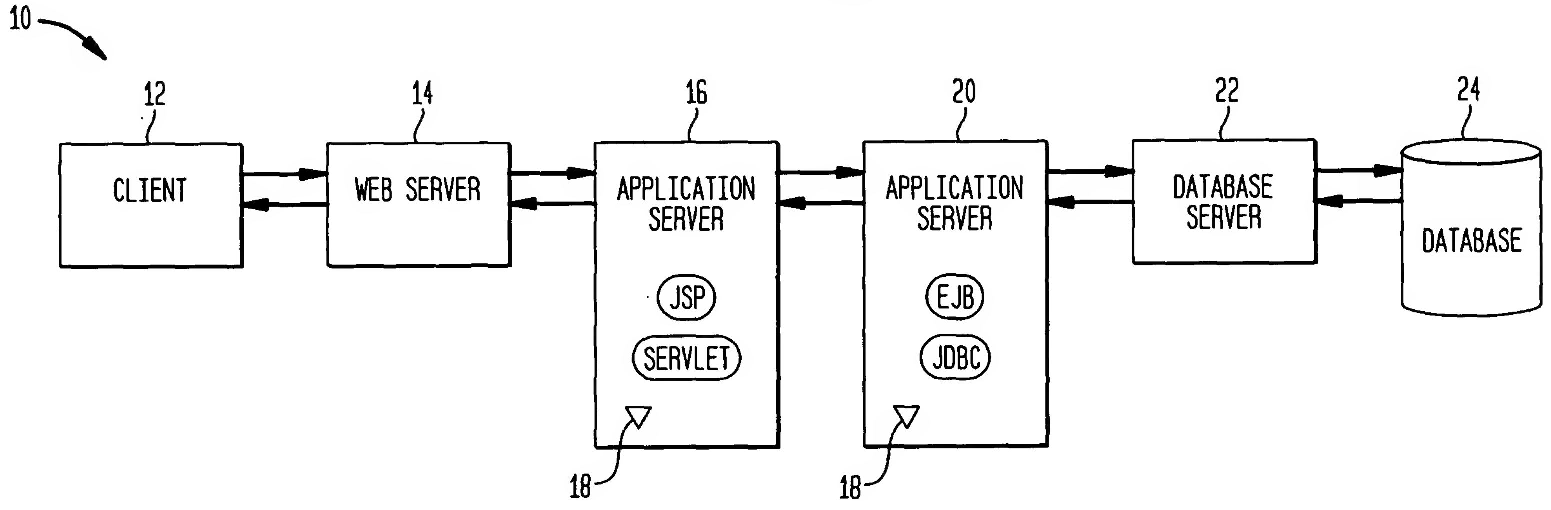

10
12
CLIENT
14
WEB SERVER
16
APPLICATION SERVER
JSP
SERVLET
18
20
APPLICATION SERVER
EJB
JDBC
18
22
DATABASE SERVER
24
DATABASE

FIG. 2

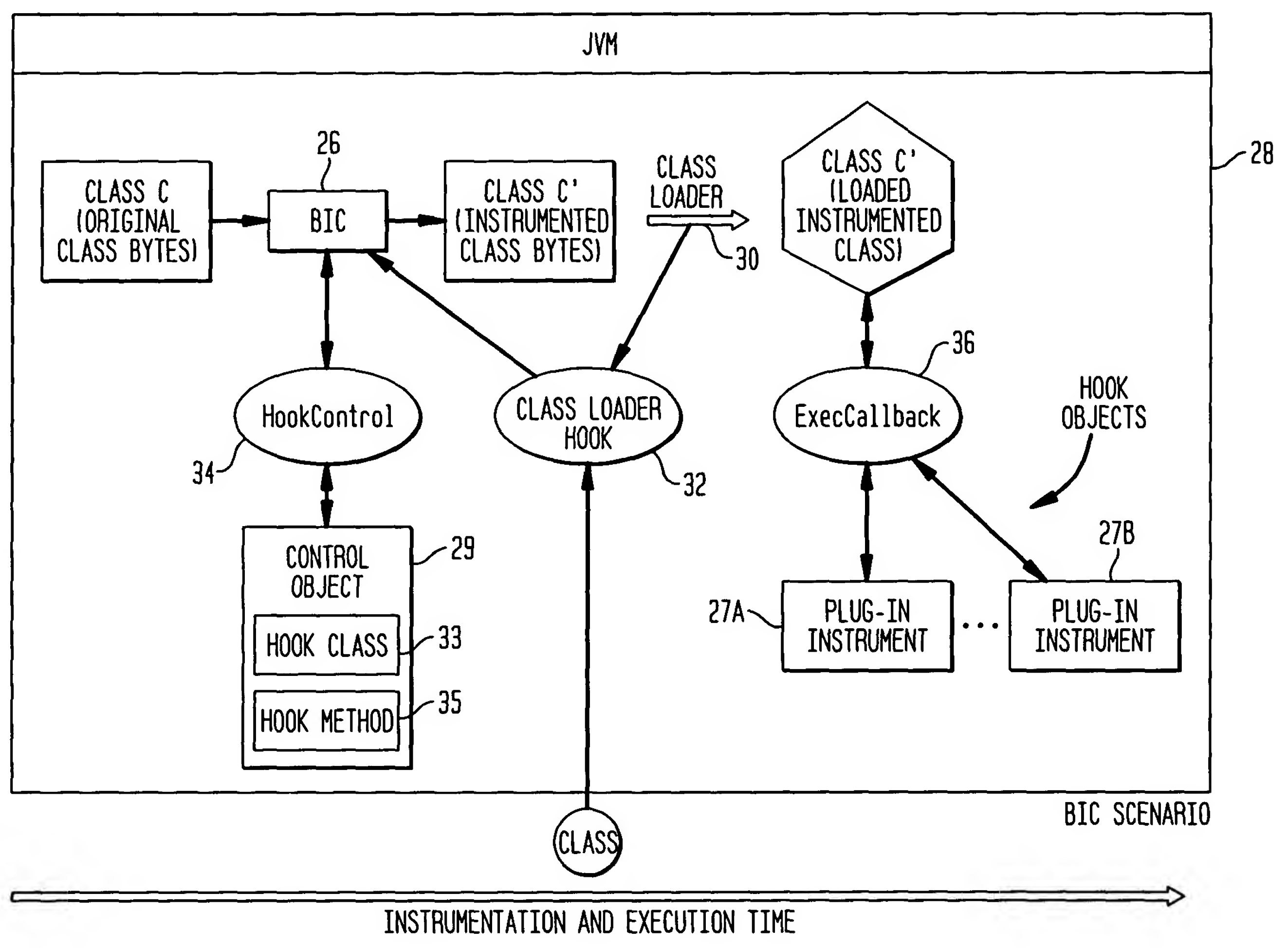

JVM
CLASS C (ORIGINAL CLASS BYTES)
26
BIC
CLASS C' (INSTRUMENTED CLASS BYTES)
CLASS LOADER
30
CLASS C' (LOADED INSTRUMENTED CLASS)
28
36
HookControl
34
CLASS LOADER HOOK
32
ExecCallback
HOOK OBJECTS
CONTROL OBJECT
29
HOOK CLASS
33
HOOK METHOD
35
27B
27A
PLUG-IN INSTRUMENT
PLUG-IN INSTRUMENT
BIC SCENARIO
CLASS
INSTRUMENTATION AND EXECUTION TIME

FIG. 3

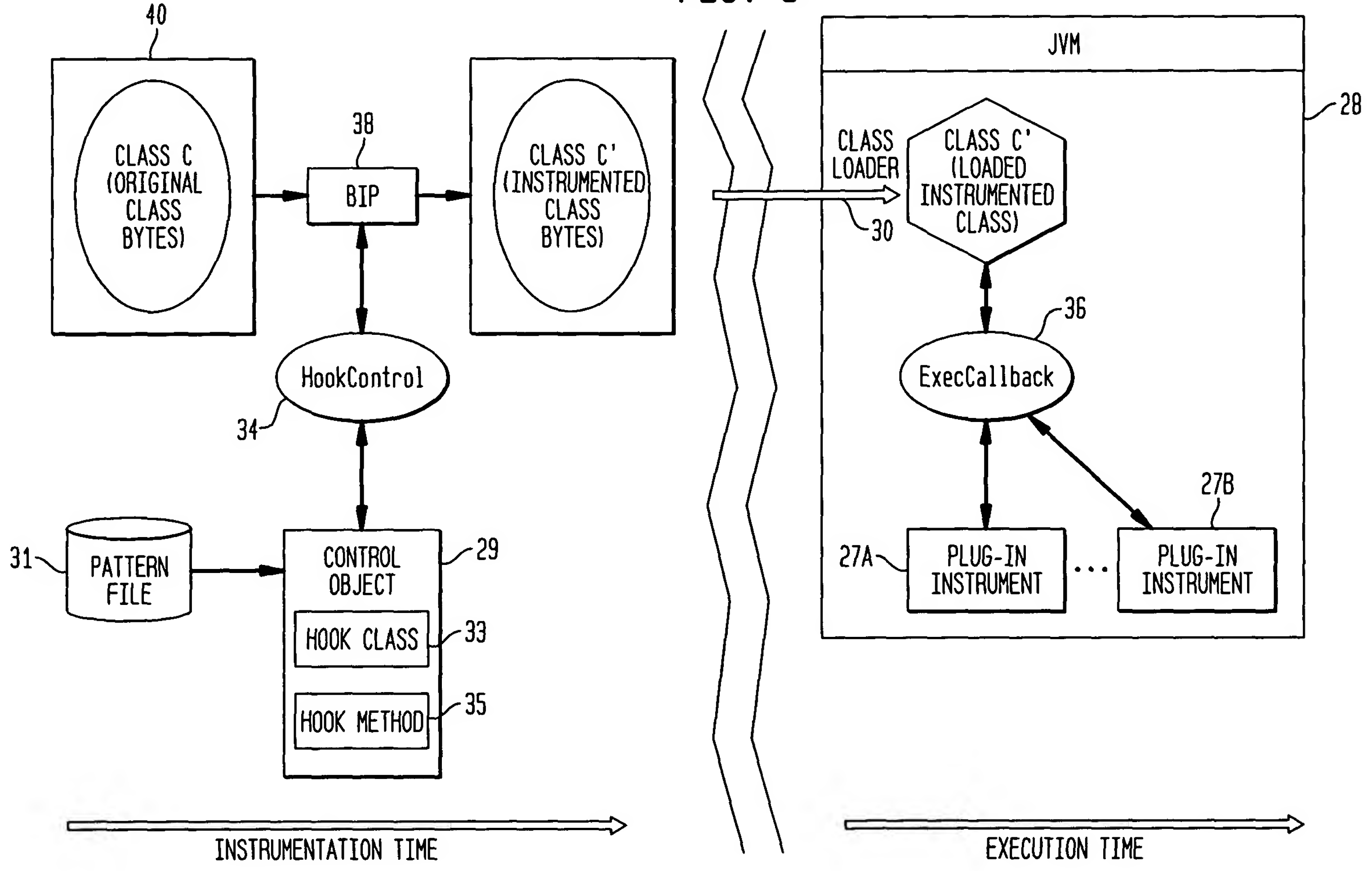
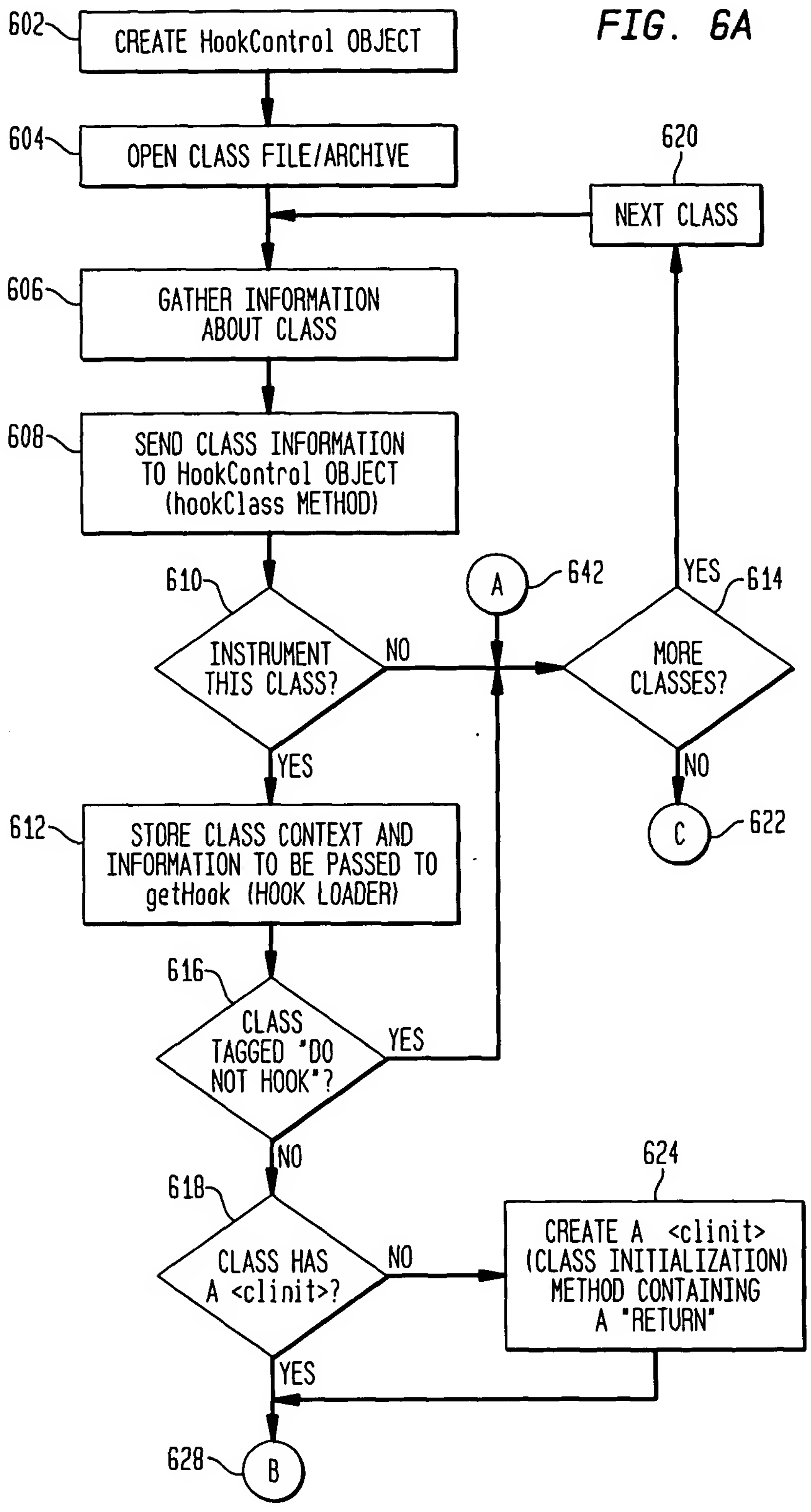

40
CLASS C (ORIGINAL CLASS BYTES)
38
BIP
CLASS C' (INSTRUMENTED CLASS BYTES)
HookControl
34
31
PATTERN FILE
CONTROL OBJECT
29
HOOK CLASS
33
HOOK METHOD
35
INSTRUMENTATION TIME
JVM
28
CLASS LOADER
30
CLASS C' (LOADED INSTRUMENTED CLASS)
36
ExecCallback
27B
27A
PLUG-IN INSTRUMENT
PLUG-IN INSTRUMENT
EXECUTION TIME

FIG. 4

```
public java.lang.Object hookClass (                              }
        java.lang.String classname,            402               }
        java.lang.String [] methods,           404               }
        java.lang.String [] superclasses,      406               } 400
        java.lang.String [] superinterfaces,   408               }
        java.lang.StringBuffer getHookArg)     410               }
```

FIG. 5

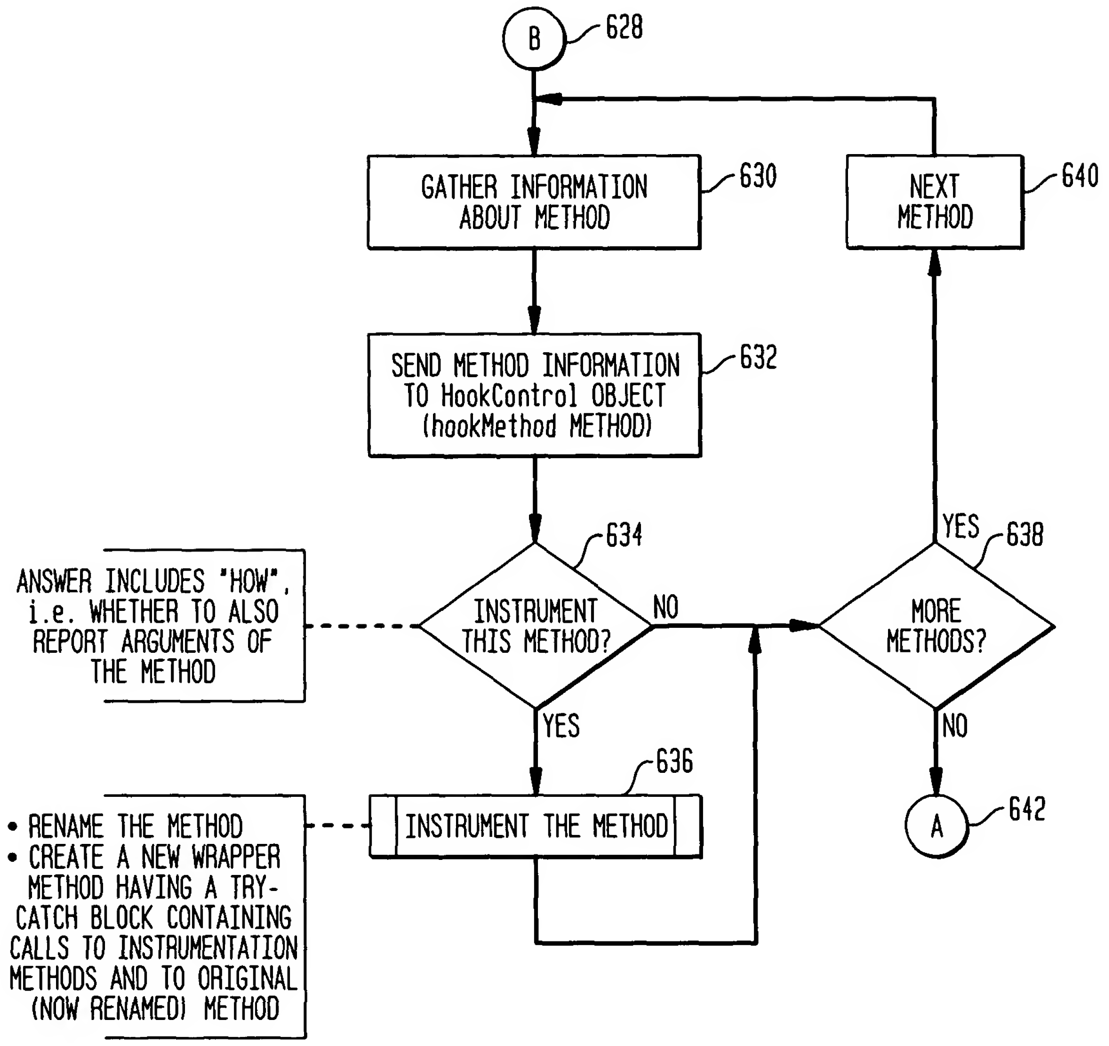
```
public int hookMethod(                                           }
        java.lang.Object classcontext,         502               }
        java.lang.String classname,            504               }
        java.lang.String methodname,           506               } 500
        java.lang.String [] superinterfaces,   508               }
        java.lang.StringBuffer defMethodArg)   510               }

public static final int DO_NOT_HOOK;           522               }
public static final int HOOK_NO_ARGS;          524               }
public static final int HOOK_WITH_ARGS;        526               } 520
public static final int HOOK_WITH_ARG1;        528               }
public static final int HOOK_WITH_ARG1_2;      530               }
public static final int HOOK_WITH_ARG2;        532               }
```



602
CREATE HookControl OBJECT
604
OPEN CLASS FILE/ARCHIVE
620
NEXT CLASS
606
GATHER INFORMATION ABOUT CLASS
608
SEND CLASS INFORMATION TO HookControl OBJECT (hookClass METHOD)
610
INSTRUMENT THIS CLASS?
NO
YES
A
642
YES
614
MORE CLASSES?
NO
C
622
612
STORE CLASS CONTEXT AND INFORMATION TO BE PASSED TO getHook (HOOK LOADER)
616
CLASS TAGGED "DO NOT HOOK"?
YES
NO
618
CLASS HAS A <clinit>?
NO
624
CREATE A <clinit> (CLASS INITIALIZATION) METHOD CONTAINING A "RETURN"
YES
628
B

FIG. 6A

FIG. 6B



B
628
GATHER INFORMATION ABOUT METHOD
630
SEND METHOD INFORMATION TO HookControl OBJECT (hookMethod METHOD)
632
INSTRUMENT THIS METHOD?
634
ANSWER INCLUDES "HOW", i.e. WHETHER TO ALSO REPORT ARGUMENTS OF THE METHOD
NO
YES
INSTRUMENT THE METHOD
636
• RENAME THE METHOD
• CREATE A NEW WRAPPER METHOD HAVING A TRY-CATCH BLOCK CONTAINING CALLS TO INSTRUMENTATION METHODS AND TO ORIGINAL (NOW RENAMED) METHOD
MORE METHODS?
638
YES
NO
NEXT METHOD
640
A
642

FIG. 6C

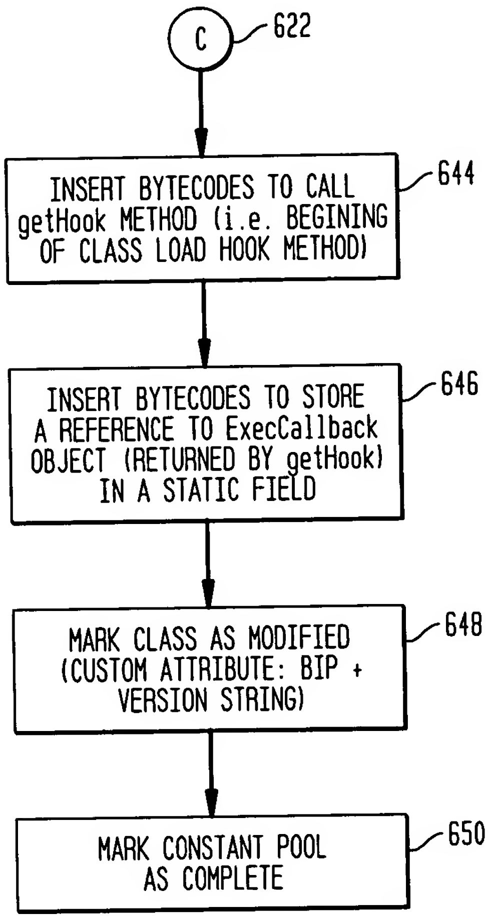

C
622
INSERT BYTECODES TO CALL getHook METHOD (i.e. BEGINING OF CLASS LOAD HOOK METHOD)
644
INSERT BYTECODES TO STORE A REFERENCE TO ExecCallback OBJECT (RETURNED BY getHook) IN A STATIC FIELD
646
MARK CLASS AS MODIFIED (CUSTOM ATTRIBUTE: BIP + VERSION STRING)
648
MARK CONSTANT POOL AS COMPLETE
650

FIG. 7

```
     public TradeResult buy(String string, int i)                                          ┐
     {                                                                                     │
      Object object; 728                                                                   │
      Throwable throwable;                                                                 │
      TradeResult tradeResult; 718                                                         │
734   if ($BIP$hook = null)                                                                │
           $BIP$installHook();  702                                                        │
726   object=$BIP$hook.methodEntry($BIP$ref_C,$BIP$ref_M0,this,2);                         │
730   if (object!=null)                                                                    │
      {                  708                                                               │
            $BIP$hook.reportArg(object,$BIP$ref_C,$BIP$ref_M0,1,string);                   │
            $BIP$hook.reportArg(object,$BIP$ref_C,$BIP$ref_M0,2,i);                        │
      }                                                                                    ├700
    ┌ try   720  710      704                                                              │
714 ┤ {                                                                                    │
    │       tradeResult = $BIP$buy(string, i);                                             │
    └ }                         712                                                        │
      catch (Throwable throwable)                                                          │
      {              716                                                                   │
            $BIP$hook.methodException(object,$BIP$ref_C,$BIP$ref_M0,throwable);            │
            throw throwable;                                                               │
      }                                                                                    │
732   if (object!=null)      706                                                           │
            $BIP$hook.methodExit(object,$BIP$ref_C,$BIP$ref_M0,tradeResult);               │
      return tradeResult;                                                                  ┘
     }                 722                                                                 ┐
     ...  724                                                                              │
     private TradeResult $BIP$buy(String string,int i)                                     ├701
      ... Original, unmodified conents of buy                                              │
     }                                                                                     ┘
```

FIG. 8A

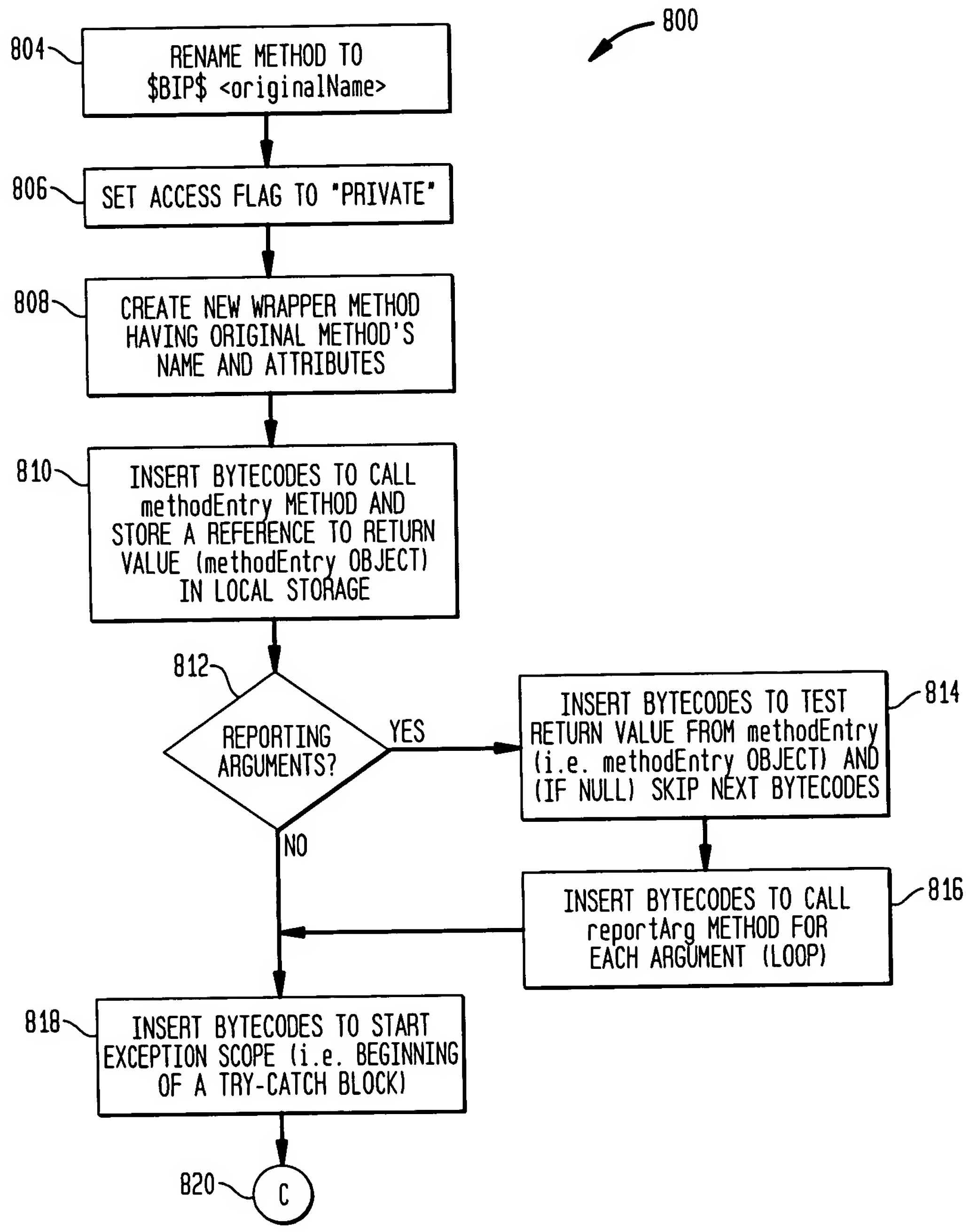
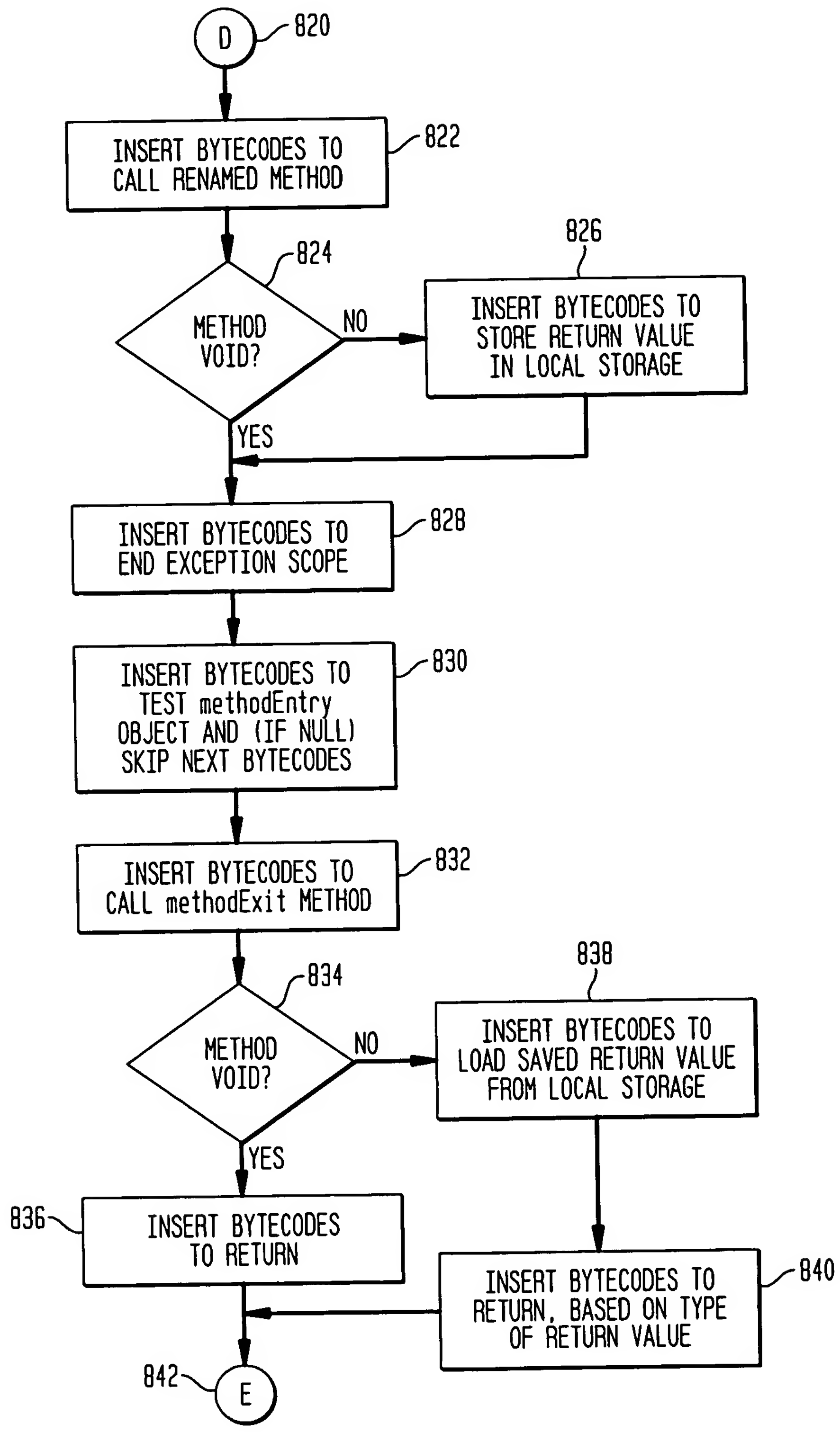

800
804
RENAME METHOD TO BIP <originalName>
806
SET ACCESS FLAG TO "PRIVATE"
808
CREATE NEW WRAPPER METHOD HAVING ORIGINAL METHOD'S NAME AND ATTRIBUTES
810
INSERT BYTECODES TO CALL methodEntry METHOD AND STORE A REFERENCE TO RETURN VALUE (methodEntry OBJECT) IN LOCAL STORAGE
812
REPORTING ARGUMENTS?
YES
NO
814
INSERT BYTECODES TO TEST RETURN VALUE FROM methodEntry (i.e. methodEntry OBJECT) AND (IF NULL) SKIP NEXT BYTECODES
816
INSERT BYTECODES TO CALL reportArg METHOD FOR EACH ARGUMENT (LOOP)
818
INSERT BYTECODES TO START EXCEPTION SCOPE (i.e. BEGINNING OF A TRY-CATCH BLOCK)
820
C

FIG. 8B



D
820
INSERT BYTECODES TO CALL RENAMED METHOD
822
824
METHOD VOID?
NO
YES
826
INSERT BYTECODES TO STORE RETURN VALUE IN LOCAL STORAGE
INSERT BYTECODES TO END EXCEPTION SCOPE
828
INSERT BYTECODES TO TEST methodEntry OBJECT AND (IF NULL) SKIP NEXT BYTECODES
830
INSERT BYTECODES TO CALL methodExit METHOD
832
834
METHOD VOID?
NO
YES
838
INSERT BYTECODES TO LOAD SAVED RETURN VALUE FROM LOCAL STORAGE
836
INSERT BYTECODES TO RETURN
INSERT BYTECODES TO RETURN, BASED ON TYPE OF RETURN VALUE
840
842
E

FIG. 8C

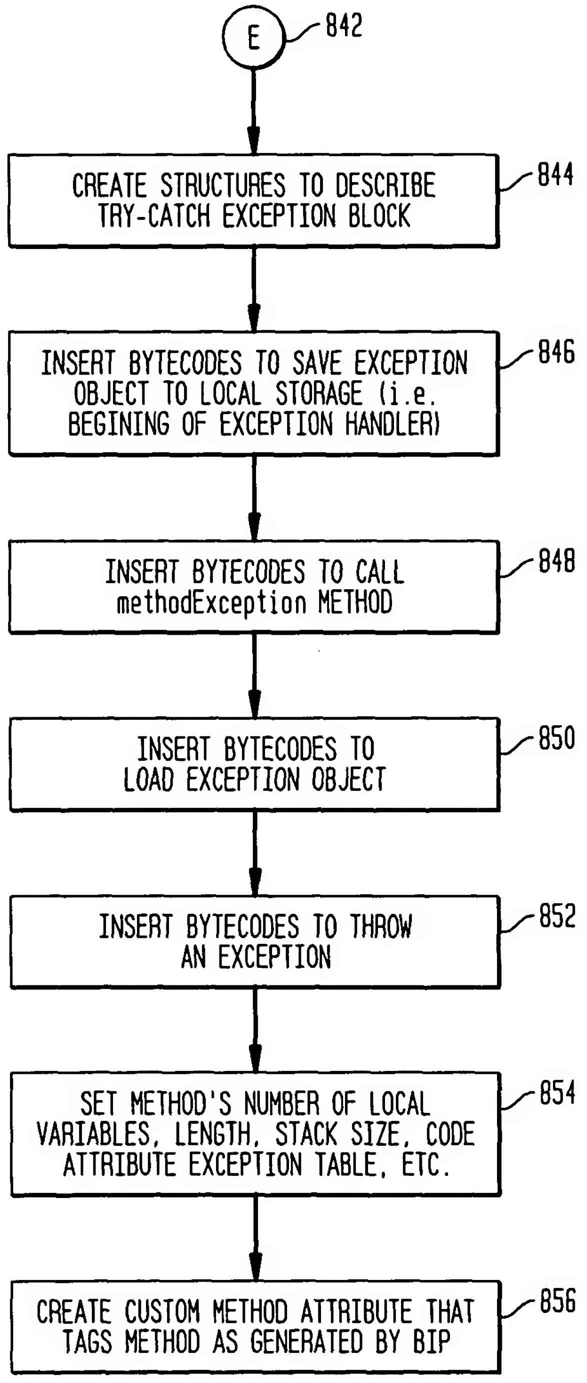

E
842
CREATE STRUCTURES TO DESCRIBE TRY-CATCH EXCEPTION BLOCK
844
INSERT BYTECODES TO SAVE EXCEPTION OBJECT TO LOCAL STORAGE (i.e. BEGINING OF EXCEPTION HANDLER)
846
INSERT BYTECODES TO CALL methodException METHOD
848
INSERT BYTECODES TO LOAD EXCEPTION OBJECT
850
INSERT BYTECODES TO THROW AN EXCEPTION
852
SET METHOD'S NUMBER OF LOCAL VARIABLES, LENGTH, STACK SIZE, CODE ATTRIBUTE EXCEPTION TABLE, ETC.
854
CREATE CUSTOM METHOD ATTRIBUTE THAT TAGS METHOD AS GENERATED BY BIP
856

FIG. 9

```
public java.lang.Object classLoadStart (                     ⎫
       java.lang.String   classname,   902                   ⎬ 900
       java.lang.class    classObj,    904                   ⎪
       int methods)   906                                    ⎭

public java.lang.Object defMethod (                          ⎫
       java.lang.Object   classref,    922                   ⎬ 920
       java.lang.String   methodname,  924                   ⎪
       java.lang.String   methodkind)  926                   ⎭

public void classLoadEnd(                                    ⎫ 940
       java.lang.Object   classref)    942                   ⎭
```

FIG. 10

```
public java.lang.Object   methodEntry (                      ⎫
       java.lang.Object   classref,    1002                  ⎪
       java.lang.Object   methodref,   1004                  ⎬ 1000
       java.lang.Object   instance,    1006                  ⎪
       int args)   1008                                      ⎭

public void reportArg (                                      ⎫
       java.lang.Object   context,     1022                  ⎪
       java.lang.Object   classref,    1024                  ⎬ 1020
       java.lang.Object   methodref,   1026                  ⎪
       int argNumber,  1028                                  ⎪
       java.lang.Object   methodArg)   1030                  ⎭

public void methodExit (                                     ⎫
       java.lang.Object   context,     1042                  ⎪
       java.lang.Object   classref,    1044                  ⎬ 1040
       java.lang.Object   methodref,   1046                  ⎪
       java.lang.Object   result)      1048                  ⎭
```

FIG. 11

```
public java.lang.Object   methodEntryOneArg(            }
       java.lang.Object   classref,                     }
       java.lang.Object   methodref,                    } 1100
       java.lang.Object   instance,                     }
       java.lang.Object   selectedArg) ~ 1102           }

public void methodException (                           }
       java.lang.Object   context,                      }
       java.lang.Object   classref,                     } 1120
       java.lang.Object   methodref,                    }
       java.lang.Throwable e) ~ 1122                    }
```

FIG. 12

```
public static ExecCallback getHook ( ~ 1202                     }
       java.lang.String   className, ~ 1204                     }
       java.lang.String   classKind, ~ 1206                     }
       java.lang.String   className, ~ 1208                     } 1200
       java.lang.String   classVersion, ~ 1210                  }
       java.lang.String   interface Version)                    }
```

FIG. 13A

1300

```
// $Source: /data1/nebula/ccm/jade/ccm/import/arra_jlink/i2/bip/hook/RCS/NullExec?Callback.java,v $
// $Revision: 1.8 $ $Date: 2001/08/28 14:56:29 $ $Author: arav $
package i2.bip.hook;
/** An implementation of the ExecCallback that does nothing.
 * A suitable base class for a custom hook class.
 */
public class NullExecCallback
     // Explicit DoNotHook for BIC testing
     implements ExecCallback, DoNotHook {

     // Called at start of class initialization
     // Returns opaque class ref
     public Object classLoadStart(String classname, Class classObj, int methods)
     {
       return null;
     }

     // Called once for each instrumented method in the class.
     // Returns opaque method ref
     public Object defMethod(
       Object classref,
       String methodname,
       String methodkind)
     {
       return null;
     }

     // End of class initialization instrumentation
     public void classLoadEnd(Object classref) { }

     // Called at instrumented method entry.
     public Object methodEntry(
       Object classref,
       Object methodref,
       Object instance,
       int args)
     {
       return null;    // Disables methodExit & reportArg instrumentation
     }

     // Called at instrumented method entry when single arg requested.
```

FIG. 13B

1300

```
public Object methodEntryOneArg(
  Object classref,
  Object methodref,
  Object instance,
  Object selectedArg)
{
  return null;     // Disables methodExit & reportArg instrumentation
}

public Object methodEntryOneTwoArg(
  Object classref,
  Object methodref,
  Object instance,
  Object arg1,
  Object arg2)
{
  return null;     // Disables methodExit & reportArg instrumentation
}

// Called at normal instrumented method exit,
// unless returned methodEntry context is null.
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  Object result) { }

// Overloaded versions of methodExit for primitive return types.
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  int result) { }         // Covers boolean, byte, char, short, and int
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  float result) { }
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
```

FIG. 13C

1300

```
  long result) { }
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  double result) { }
public void methodExit(
  Object context,
  Object classref,
  Object methodref) { }

// Called unconditionally at method exception
public void methodException(
  Object context,
  Object classref,
  Object methodref,
  Throwable e) { }

//-------------------
// Argument reporting
//-------------------

// Called after instrumented method entry, once per arg, if
// argument reporting was instrumented.
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  Object methodArg)       // The actual argument (reference types)
{
}

// Overloaded versions of reportArg for primitive types.
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  int methodArg)  // Covers boolean, byte, char, short, and int
{
}
```

FIG. 13D

1300

```
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  float methodArg)
{
}
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  long methodArg)
{
}
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  double methodArg)
{
}

}   // class NullExecCallback
```

FIG. 14

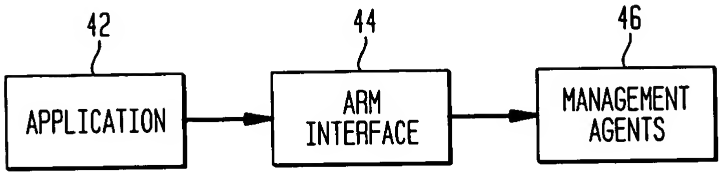


FIG. 15

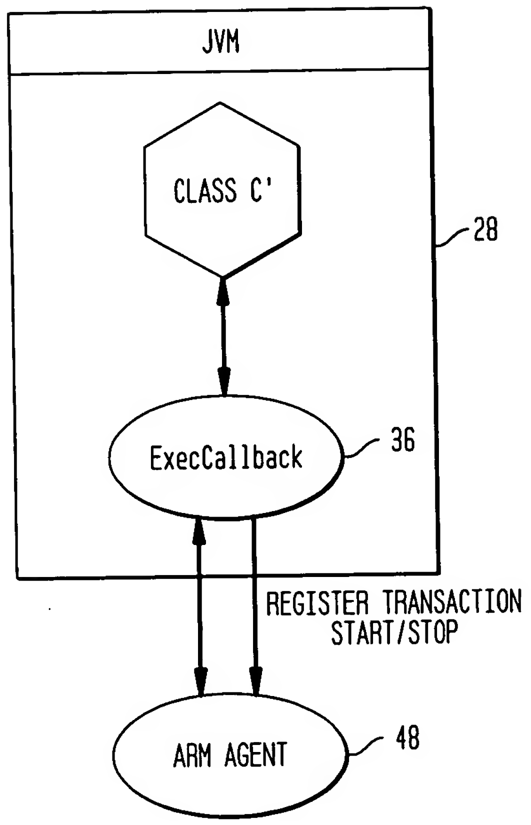
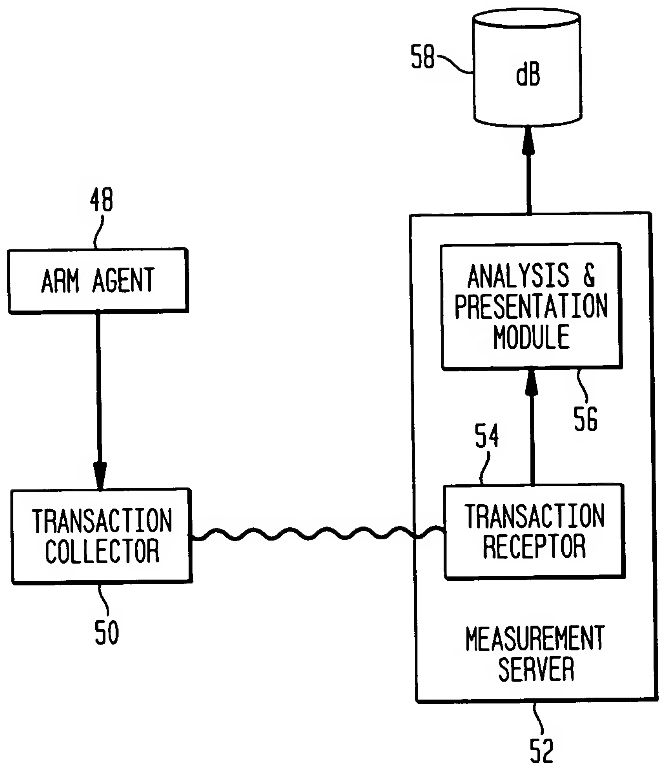

JVM
CLASS C'
28
ExecCallback
36
REGISTER TRANSACTION START/STOP
ARM AGENT
48

FIG. 16



58
dB
48
ARM AGENT
ANALYSIS & PRESENTATION MODULE
56
54
TRANSACTION COLLECTOR
TRANSACTION RECEPTOR
50
MEASUREMENT SERVER
52

FIG. 17

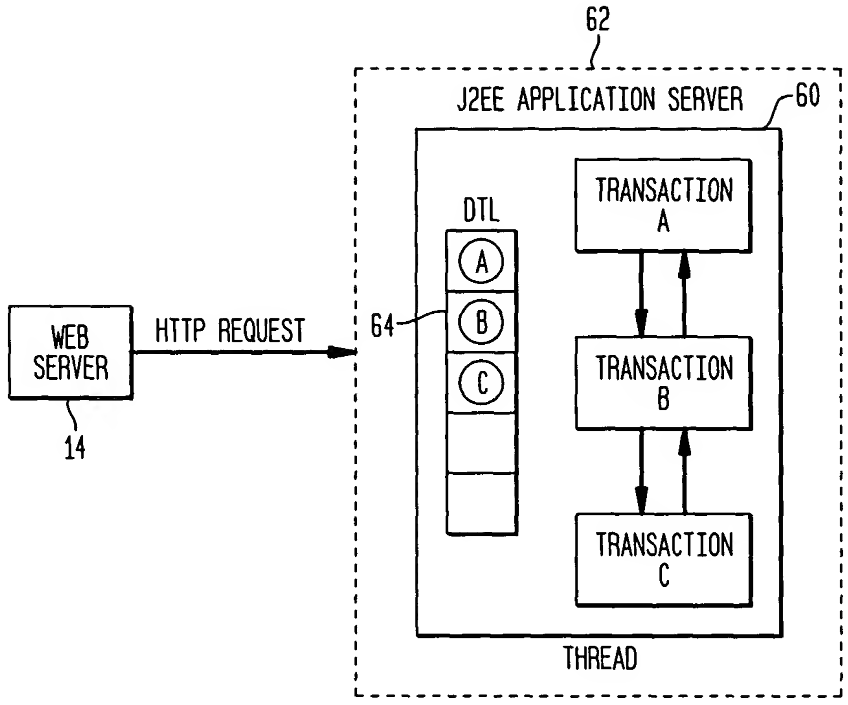
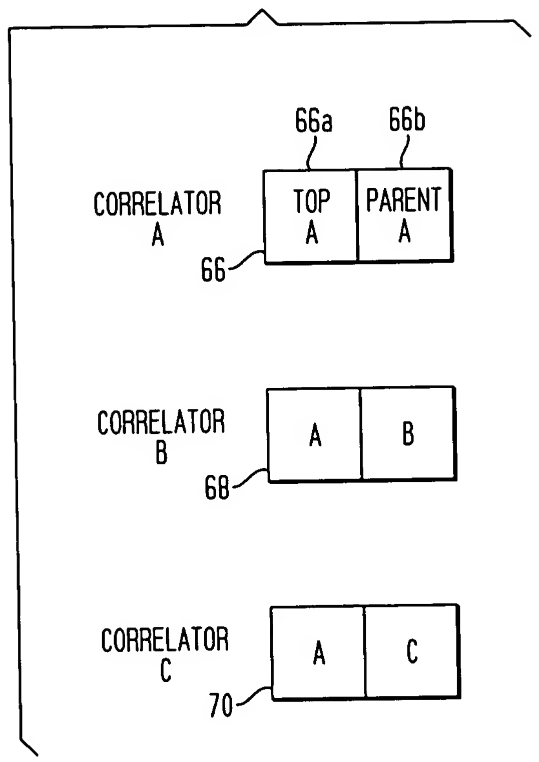
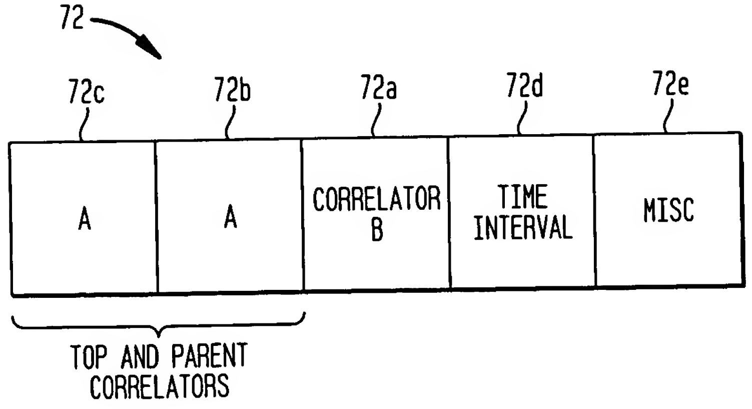

62
J2EE APPLICATION SERVER
60
DTL
A
B
C
64
WEB SERVER
14
HTTP REQUEST
TRANSACTION A
TRANSACTION B
TRANSACTION C
THREAD

FIG. 18



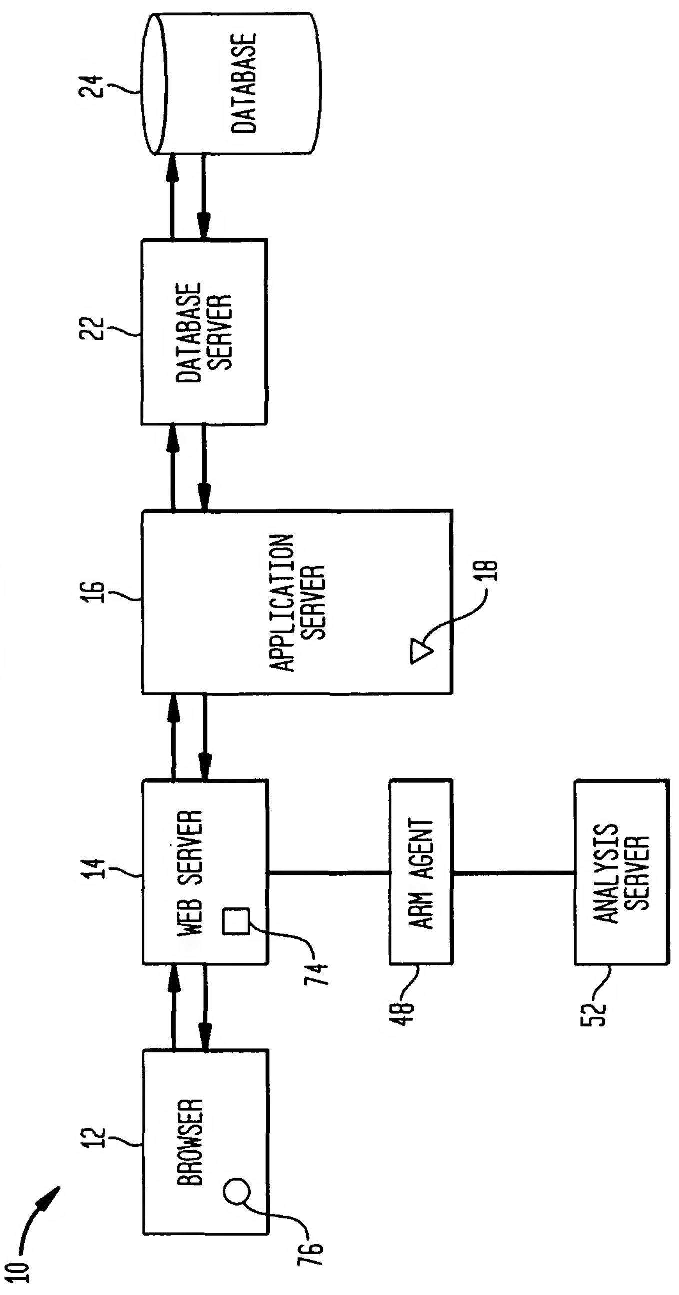
CORRELATOR A
66
66a
66b
TOP A
PARENT A
CORRELATOR B
68
A
B
CORRELATOR C
70
A
C

FIG. 19



72
72c
72b
72a
72d
72e
A
A
CORRELATOR B
TIME INTERVAL
MISC
TOP AND PARENT CORRELATORS

FIG. 20


10
12
BROWSER
76
14
WEB SERVER
74
16
APPLICATION SERVER
18
22
DATABASE SERVER
24
DATABASE
48
ARM AGENT
52
ANALYSIS SERVER

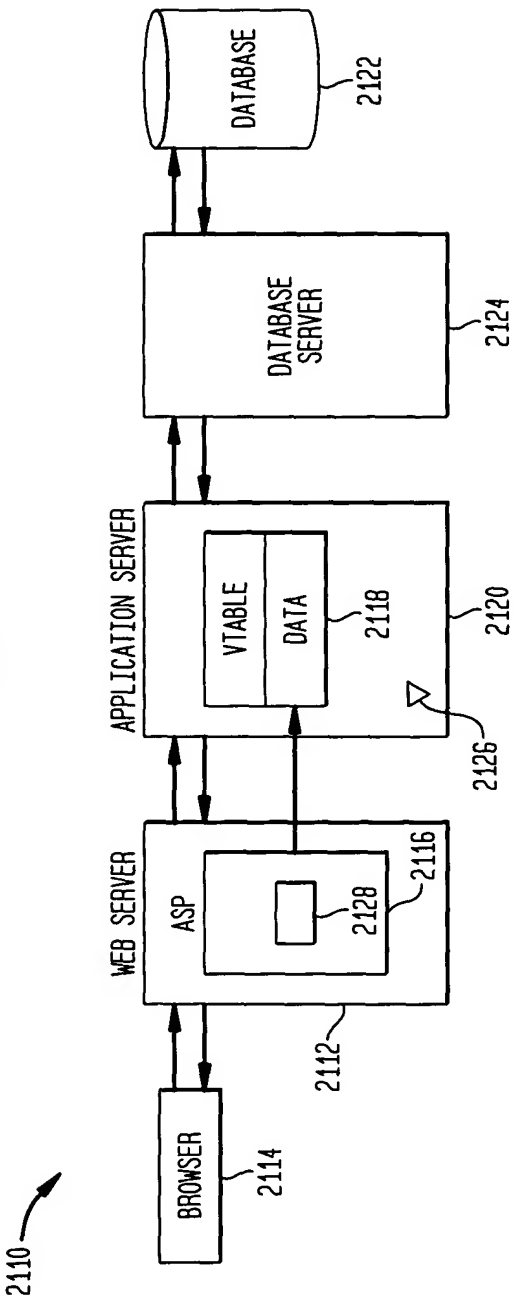

2110
BROWSER
2114
WEB SERVER
ASP
2128
2116
2112
APPLICATION SERVER
VTABLE
DATA
2118
2120
2126
DATABASE SERVER
2124
DATABASE
2122

FIG. 21

FIG. 22

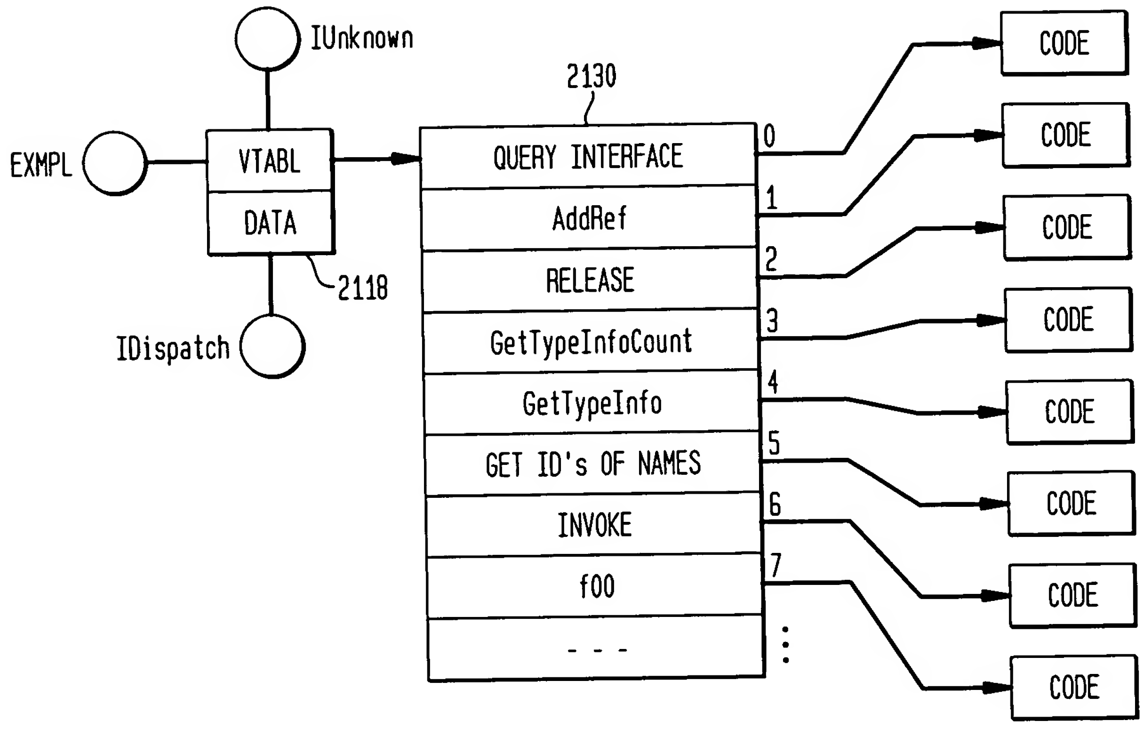

IUnknown
2130
EXMPL
VTABL
DATA
2118
IDispatch
QUERY INTERFACE
AddRef
RELEASE
GetTypeInfoCount
GetTypeInfo
GET ID's OF NAMES
INVOKE
f00
- - -
0
1
2
3
4
5
6
7
CODE
CODE
CODE
CODE
CODE
CODE
CODE
CODE

FIG. 23

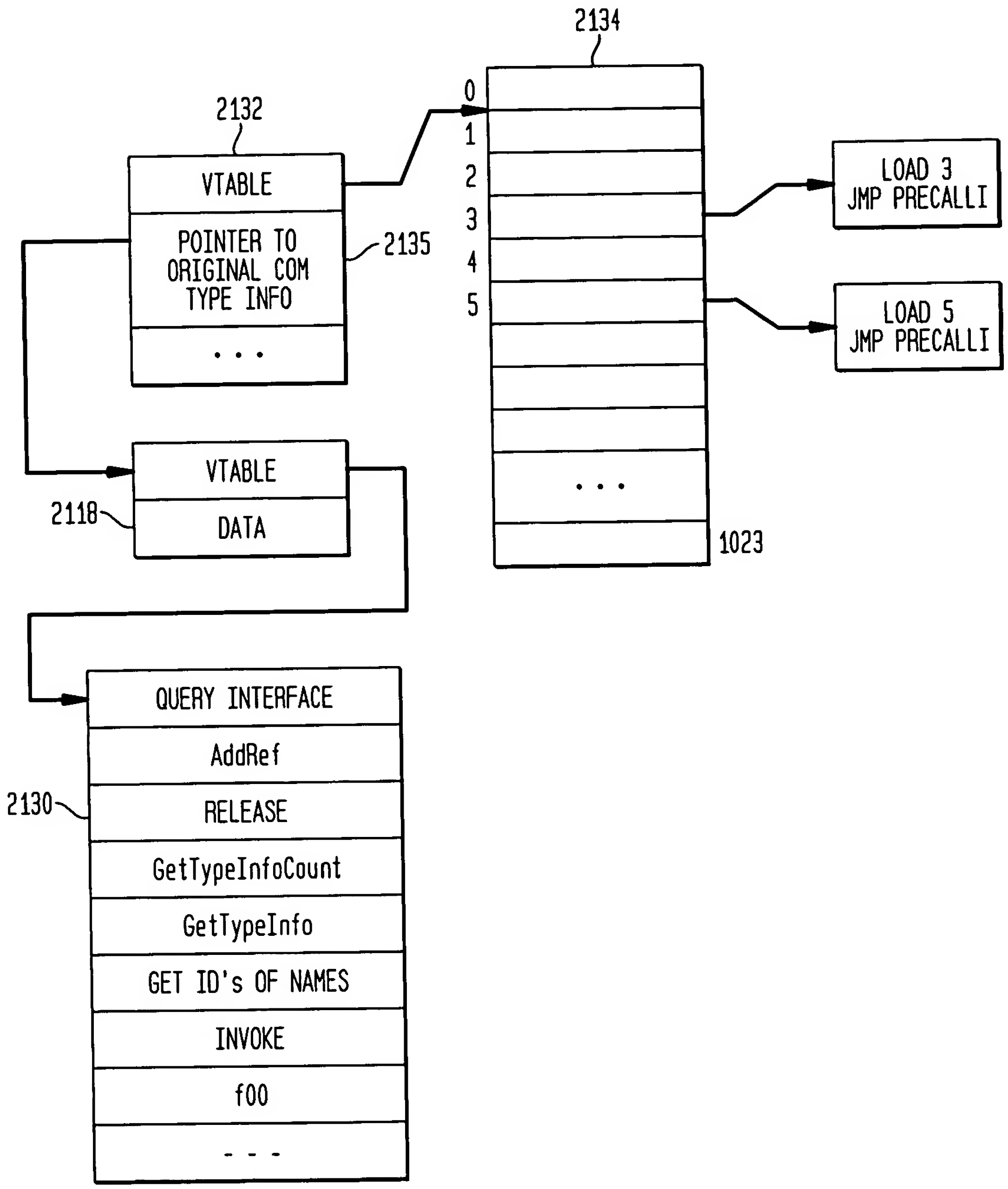

2134
0
1
2
3
4
5
1023
. . .
2132
VTABLE
POINTER TO ORIGINAL COM TYPE INFO
. . .
2135
LOAD 3 JMP PRECALLI
LOAD 5 JMP PRECALLI
VTABLE
DATA
2118
QUERY INTERFACE
AddRef
RELEASE
GetTypeInfoCount
GetTypeInfo
GET ID's OF NAMES
INVOKE
f00
- - -
2130

FIG. 24

```
PrecallInterceptor UNIVERSAL COM METHOD (METHOD #){
          DETERMINE ARGUMENTS NEEDED FOR METHOD #
          ARM START
          CALL ORIGINAL METHOD #
          ARM STOP
          }
```

FIG. 25

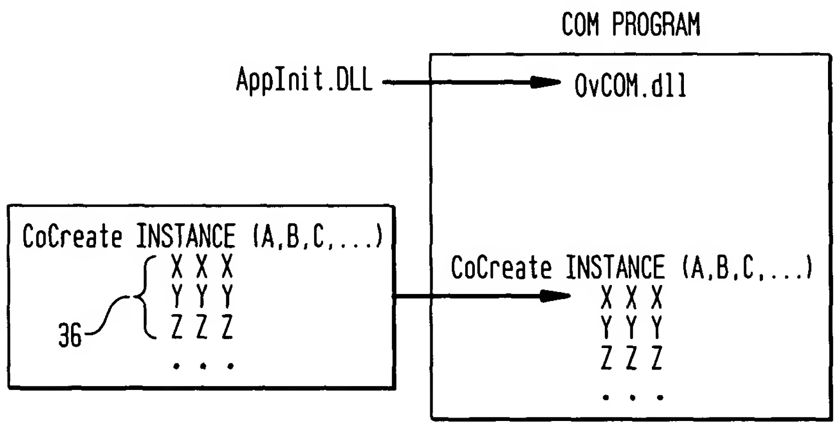


FIG. 26

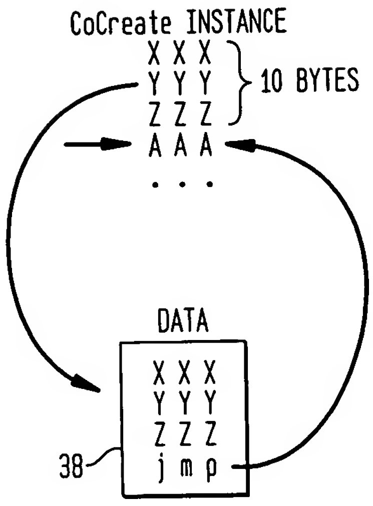


FIG. 27

```
OVTA CoCreateInstance (A,B,C) {
        .
        .
        .
    CALL CoCreateInstance (A,B,C) {
        .
        .
        .
        ACCESS B
        WRAP OBJECT REFERRED BY B
        SET B TO POINT TO WRAPPER OBJECT
        RETURN TO ORIGINAL CALLER
        }
```

FIG. 28

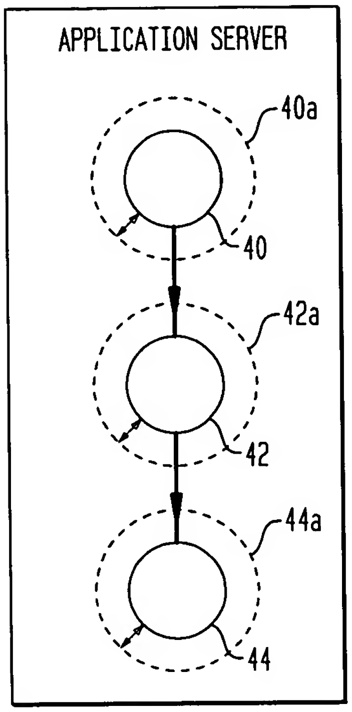
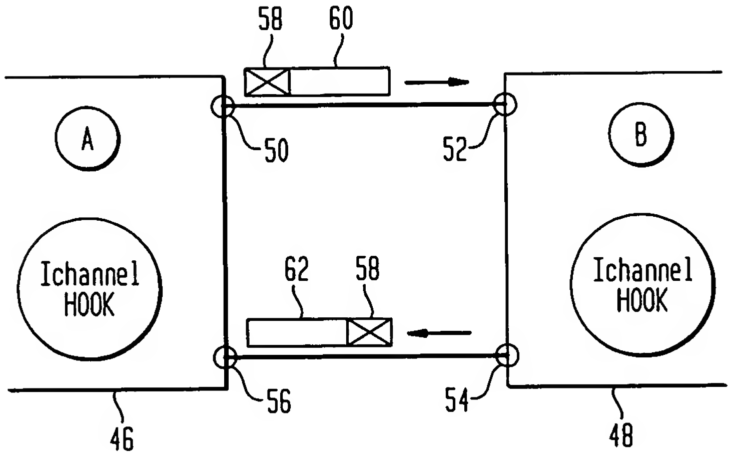

APPLICATION SERVER
40a
40
42a
42
44a
44

FIG. 29


58
60
A
B
50
52
Ichannel HOOK
Ichannel HOOK
62
58
56
54
46
48